as shown in figure 2. After executing these 3 techniques, results into slot utilization and utilization efficiency optimization. It Also improves the data locality and load balancing.

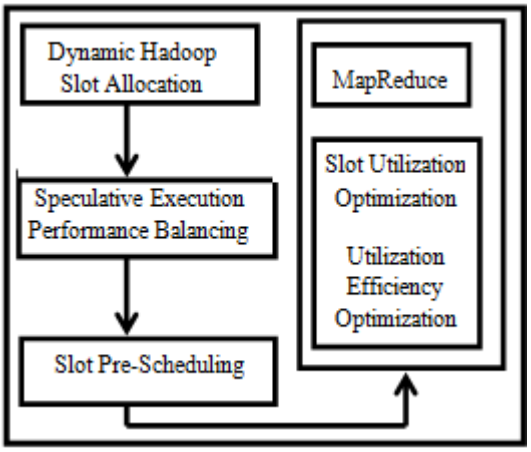


Figure 2: Overview of the Proposed System

Dynamic Hadoop Slot Allocation

It attempt to maximize slot utilization while maintainingthe fairness, when there are pending tasks (e.g., map tasks orreduce tasks). We break implicit assumption of MapReducethat the map tasks can only run on map slots & reduce taskscan only run on reduce slots. In our proposed system wemodify it that map and reduce tasks can berun on either mapor reduce slots.

There are 4 cases,

Consider,

NM = Total number of Map tasks

NR = Total number of Reduce tasks

SM = Total number of map slots

SR = Total number of reduce slots

Case 1: NM ≤ SMandNR ≤ SR

The map tasks which are running on map slots and reduce tasks are run on reduce slots, There is no borrowing of map and reduce slots.

Case 2: NM>SMandNR<SR

We satisfy reduce tasks for reduce slots first and then use those idle reduce slots for running map tasks.

Case 3: NM<SMandNR>SR

We can schedule those unused map slots for running reducetasks.

Case 4: NM>SMandNR>SR

The system should be in completely busy state.

1) Speculative Execution Performance Balancing:

It identifies the slot resource in-efficiency problem for aHadoop cluster, caused by speculative tasks. It works on top ofthe Hadoop speculative scheduler to balance the performancetradeoff between a single job and a batch of jobs. SlotPre-Scheduling improves the slot utilization efficiency andperformance by improving the data locality for map taskswhile keeping the fairness.Speculative tasks are competing for certain resources includingnetwork, map slots and reduce tasks. For maximizing theperformance we should complete the pending tasks first beforeconsidering the speculative tasks. When node is having idlemap slot, then we should consider pending map task firstlyand then we consider speculative map task. For an idle mapslot, we first check jobs J1, J2 ... Jn for map task.For every job we check the total number of pending map andreduce tasks by considering all jobs from Ji and Jj. Where,i=1,2,3,4....

j= i + maxNumOfJobsCheckedForPendingTasks − 1.

We checked each jobJi by considering 3 conditions: 1) Totalpending map tasks are greater than zero. 2) No failed pendingmap tasks and map tasksfor job Ji. 3) Total pending reducetasks is greater than zero.

2) Slot Pre-Scheduling

It improves the slot utilization efficiency and performance by improving the data locality for map tasks while keeping the fairness.

Step 1: Compute load factor mapSlotsLoadFactor = Pending map tasks +running map tasksfrom all jobs divided by the cluster map slot capacity.

Step 2: Compute current maximum number of usable map slots = number ofmap slots in a tasktracker * minmapSlotsLoadFactor, 1.

Step 3: Compute current allowable idle map (or reduce) slots for a tasktracker= maximum number of usable map slots - current number ofused map (or reduce) slots

D. Mathematical Model

The mathematical terminology of proposed system is explained below:

Let S be the proposed system

S = {I,O,F,Fs,F*l*,ϕ}

Identify the inputs I.

I = {T1, M, T2, R,U,E}

Where,

T1 = Pending Map Tasks.

M = Idle Map Slots.

T2 = Pending Reduce Tasks.

R = Idle Reduce Slots.

U = Utilized Slots.

E = Empty Slots.

Identify set of Function. Let F be the set of Functions.

F = {F1,F2,F3}

Where,

F1 = Verify Information.

F2 = Dynamic Slot Allocation.

F3 = Balance the Performance of Job.

Identify the Outputs. Let O be the set of outputs.

O = {O1,O2}

Where,

O1 = Slots Allocated Successfully.

O2 = Successfully Balance the Performance of Job.

Final State:

FS = Increase the Performance of Mapper & Reducer

Failure case:

F*l*= Errors in measuring the input parameters

Constraints:

Let ϕ be the constraints ϕ =C*l*

Where,

C*l* = Accuracy in measuring the input parameters.

E. Experiments and Result

In This section we have shown the working of the proposed system. The figure 3 shows the total required time for the completion of task for proposed system in contrast with the existing system which is Maximum Cost Performance. In figure 3 it also shows the specific time required for all the three techniques Dynamic Hadoop Slot Allocation (DHSA),

Speculative Execution Performance Balancing (SEPB) and Slot Pre-Scheduling.

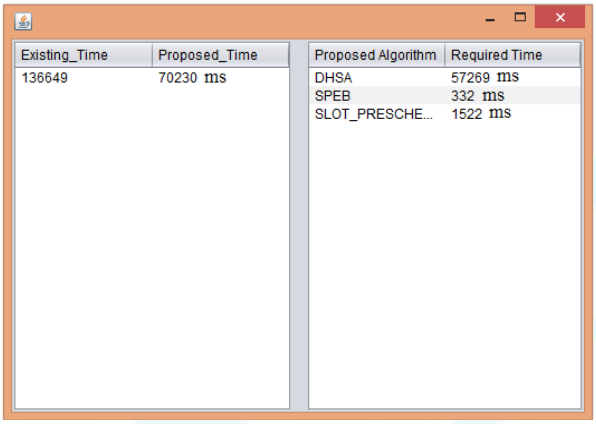
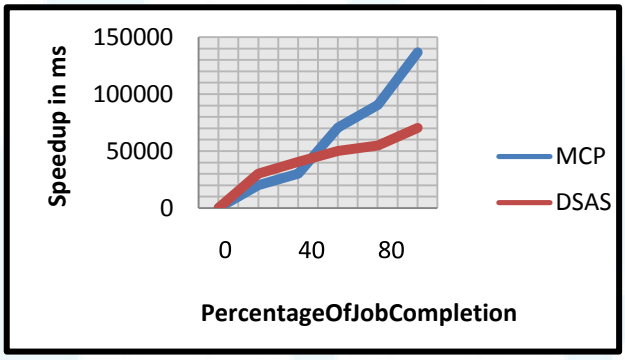


Figure 3: Required time for the proposed system

The graph 1 shows the time required to complete the tasks in MCP is higher as compared to DSAS. The performance of the MCP degrades as the time speeds up.



Graph 1: Performance improvement of the system

4. Conclusion

The aim of the proposed system is to improve the performance of MapReduce workloads. It considered three techniques: Dynamic Hadoop Slot Allocation, Speculative Execution Performance Balancing, and Slot Pre-Scheduling. Dynamic Hadoop Slot Allocation uses allocation of map to maximize the slot utilization and it reduces the task dynamically. It does not require any prior information or any assumption and it can be run on any kind of MapReduce jobs. Speculative Execution Performance Balancing identifies the slot inefficiency problem. It manages the balance between single and batch of jobs dynamically. Slot Pre-Scheduling are used to enhance the efficiency of slot utilization by maximizingdata locality. We can enhance the utilization by adding above concept in traditional system. In future we plan to implement above mentioned concept in cloud environment.

5. Acknowledgement

We thank all the anonymous reviewers and editors for their valuable comments and suggestions to improve the quality of this manuscript.

References

[1] Apache Hadoop. http://hadoop.apache.org.
[2] Hadoop Distributed File System, http://hadoop.apache.org/hdfs
[3] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. OSDI '04, pages 137150, 2004
[4] B. Moseley, A. Dasgupta, R. Kumar, T. Sarl, On scheduling in map-reduce and flow-shops. In SPAA'11, pp. 289-298, 2011.
[5] A. Verma, L. Cherkasova, R.H. Campbell, Orchestrating an Ensemble of MapReduce Jobs for Minimizing Their Makespan, IEEE Transaction on dependency and secure computing, 2013.
[6] A. Verma, L. Cherkasova, R. Campbell. Two Sides of a Coin: Optimizing the Schedule of MapReduce Jobs to Minimize Their Makespan and Improve Cluster Performance. In IEEE MASCOTS, pp. 11-18, 2012.
[7] S.J. Tang, B.S. Lee, and B.S. He. MROrder: Flexible Job Ordering Optimization for Online MapReduce Workloads. In Euro-Par'13, pp. 291-304, 2013.
[8] S.J. Tang, B.S. Lee, R. Fan and B.S. He. Dynamic Job Ordering and Slot Configurations for MapReduce Workloads, CORR (Technical Report), 2013.
[9] C. Oguz, M.F. Ercan, Scheduling multiprocessor tasks in a twostage flow-shop environment. Proceedings of the 21st international conference on Computers and industrial engineering, pp. 269-272, 1997.
[10] H. Herodotou, H. Lim, G. Luo, N. Borisov, L. Dong, F. B. Cetin, and S. Babu. Starfish: A Self-tuning System for Big Data Analytics. In CIDR11, pp. 261C272, 2011.
[11] J. Polo, C. Castillo, D. Carrera, et al. Resource aware Adaptive Scheduling for MapReduce Clusters. In Middleware'11, pp. 187- 207, 2011.
[12] Apache Hadoop NextGen MapReduce (YARN). http://hadoop.apache.org/docs/current/hadoop yarn/hadoopyarn- site/YARN.html.
[13] M. Zaharia, A. Konwinski , A.D. Joseph , R. Katz , I. Stoica,Improving MapReduce performance in heterogeneous environments.In OSDI'08, pp.29-42, 2008.
[14] Z.H. Guo, G. Fox, M. Zhou, Y. Ruan.Improving ResourceUtilization in MapReduce. In IEEE Cluster12. pp. 402-410,2012.
[15] G. Ananthanarayanan, S. Kandula, A. Greenberg, I. Stoica, Y.Lu, B. Saha, and E. Harris, Reining in the outliers in map-reduceclusters using mantri, in OSDI10, pp. 1-16, 2010.
[16] Q. Chen, C. Liu, Z. Xiao, Improving MapReduce PerformanceUsing Smart Speculative Execution Strategy. IEEE Transactions on Computer, 2013.
[17] M. Zaharia, D. Borthakur, J. Sarma, K. Elmeleegy,S. Schenker,I.Stoica, Delay scheduling: A simple technique for achievinglocality and fairness in cluster scheduling. In EuroSys10, pp.265-278, 2010.
[18] J. Tan, S. C. Meng, X. Q. Meng, L. Zhang. Improving ReduceTaskdata locality for sequential MapReduce jobs. In IEEEInfocom13, pp. 1627-1635, 2013.
[19] Z. H. Guo, G. Fox, and M. Zhou. Investigation of Data Localityin MapReduce. In IEEE/ACM CCGrid12, pp, 419-426, 2012.

[20] B. Palanisamy, A. Singh, L. Liu and B. Jain, Purlieus: LocalityawareResource Allocation for MapReduce in a Cloud, InSC11, pp. 1-11, 2011.
[21] Z. H. Guo, G. Fox, and M. Zhou.Investigation of data locality and fairness in MapReduce. In MapReduce12, pp, 25-32, 2012.
[22] M. Hammoud and M. F. Sakr. Locality-Aware Reduce TaskScheduling for MapReduce. In IEEE CLOUDCOM11. pp. 570-576, 2011.
[23] M. Hammoud, M. S. Rehman, M. F. Sakr. Center-of-GravityReduce Task Scheduling to Lower MapReduce Network Traffic.In IEEE CLOUD12, pp. 49-58, 2012.